3. CRYPTOGRAPHY – THE ARTIST STUDY OF ENCRYPTION

3.1 Cryptography

It is the learning of numerical practice narrated to facets of information protection for example authentication of entity and data source, confidentiality as well as integration of data. It does not merely provide the information protection, but it is also a set of mathematical techniques.

It is the exercise and revision of covering securely to the information. Recent cryptography interconnects the subjects of mathematical studies, computing technology, and electrical technology. The cryptographical uses comprise Automatic Telling Machine cards, passwords, and e-commerce. At the very early stage it was simply the cipher text that was the nonsense data that is not read by the intruders. It was only the simple conversion of the data to secure it. As for example it was like, the original word “happy”, but after the treatment it was “ibqqz”. Now a reasoning person can easily understand the data that it is the single character forward in the English alphabet. But some of the year passes it was much complex that it is not possible for the reasoning person to understand the information.

But the computer age grows more the processors with very high speeds. They can easily find the converted data to the original one. It becomes the challenge for the cryptologists to think about much more accurate and high attach bearers algorithms must be developed by the mathematicians that the computers cannot resolve them easily. There was the use of encryption keys of small size but they are also sorted out easily by the intruders with the help of computers. Now there are so many algorithms which require the big size keys. At present there are the encryption keys of size 128 bytes, 152 bytes, and 256 bytes.

Steganography

Steganography is another way to protect the data from the intruders. The information is hided into the other images. It covers the vital information very efficiently that the intruders cannot easily recognize the available information. The terminology is taken from the Greek language. It is the art for communicating in which the availability of the communication is
completely hidden. As we compare it with the cryptography where the intruders are permitted to see or intercept or detect or modify the communicating data. In the case of this technique “Steganography”, the intruders have no knowledge about the presence of the information.

In the digital electronic globe, both these techniques especially the cryptography and the steganography are utilized to secure the information from the unauthorized persons. The techniques are very outstanding for protecting the data. But any one of them independently is not the robust one. They are breakable one. Because of this cause, it is suggested by the experts that the both should be used as the security layers for the invaluable information. The steganography can be utilized on any size of files and any type of file data format in this computer world. In today’s environment the data formats which are mostly utilized are .doc, .jpg, .text, .pdf, .mp2, .mp3, .mp4, .gif, .bmp, and .wav etc. So there is a demand to protect such data formats and the huge file sizes must be secure to communicate on the networks as well as on the desktop computers. And all the information is hidden in the other type of messages and the intruders or attackers are not aware about the data or information availability.

**Cryptographic goals**

These are the basic aims for the security providers that are as follows. These are privacy of data (data confidentiality), integrity of data, user authenticity, and non-repudiation. The first three are also very well discussed in the first chapter which is named as “Introduction”.

1. The confidentiality is also called as the privacy of the data. The applications which provide the confidentiality mean they are keeping the data or information in the reach of their permitted users. And keep away the unauthorized users. It has another name that is the secrecy of data or information. It is provided through the use of various methodologies. In it the physical security as well as the algorithmic conversions takes place.

2. Integrity of data tackles the problems of data alterations or data modifications. For ensuring the integrity of the data, we must spot the manipulations of data done by the intruders. Any type of data modification can take place like insertion of new data in between, or deletion of the exact data, or substitution of the communicated data.

3. User authenticity is the target to identify the authorize person or not. It is applied on both the entities as well as information. There must be the identification of both the communicating
partners on the communication media. The messages which are delivered through the communication media should have valid authentication regarding the source of information, time and date etc. because of this it is partitioned into two main categories as the entity authentication as well as the data source authentication.

4. Another basic goal is the non-repudiation. In it the communicating party refuses that he or she has made any communication or activity previously. Because of this refusal or denial, there is a dispute between the communicating parties. So there is always a demand to find out the solution for such situations. For instance, one person may approve the procurement of assets by the other person and latterly refuses such approvals were decided. A third party should be involved to resolve such type of disputes.

   A primary objective of cryptography is to sufficiently tackle the above mentioned four aims or goals for the cryptography. The cryptography is utilized to protect and detect the misleading and other unwanted activities by the intruders of information.

   There are so many cryptographic primitives are utilized for securing and protecting the information.

   Figure 3.1 offers a systematic classification of the cryptographic primitives believed and how these are related with each other. The diverse criteria is used to evaluate these primitives which is as follows:

   ✓ Security level: It is regularly typical to enumerate. It is always given or shown with the help of the operations counting which is needed to overcome the intentions of the unauthorized person’s motives. It is the upper bound of the work done to overcome the intender’s aims. It is also named as the “work factor”.

   ✓ Utility (Functionality): The security’s basic goals are in requirement of combining all the objectives. The effectiveness of the primitives is considered according to its achievement.
Operational Methods: When these basic objective of the cryptography are implemented in diverse modes and on the diverse inputs, than they reveal the different properties. So these basic security aims should offer diverse utilities or functionalities. It depends on the mode of the application.

Performance: This is another criterion in which the space acquired and the time taken is considered. For instance we can say how much bits or bytes are taken by the algorithm at once.

Ease of Use: In it we consider the development of particular basic aim (implementation). Implementation complexity is considered here. It may be implemented in software or in the hardware.

3.1.1 Symmetrical Cryptography
It is also known as the private cryptography. In a message or a block of bytes is enciphered and deciphered with the help of the private key. In the Figure 3.2 a symmetric cryptosystem is shown.

![Figure 3.2: A symmetric cryptosystem.](image)

It is a well known cryptography technique and it the early age cryptography. It is utilized from the very start of the cryptography and at present also. It has a number of compensations which are as follows:

- Symmetrical ciphers are planned to have high speed of throughput. So many hardware implementations are achieving high speed encryptions just at the speed of hundred of megabytes of data per second. But the software implementations are achieving fewer throughputs as compared to the hardware.
- Short keys are utilized in this type of encryption.
- Symmetrical ciphers are utilized for generate pseudo number generators as well as hash functions, and computational digital signature schemes etc.
- Symmetrical ciphers are utilized to generate much muscular ciphers. Easy conversions that are easily analyzed. But the muscular and strong product are constructed.
- Symmetrical enciphering process is historical in nature. It was utilized from the old age we can say that it was used before the discovery of computer machines.

It has some disadvantages also like, in the two way communication then the privte key is utilize on both the sides.

There are so many popularly used symmetric key algorithms like DES, and tripleDES, Rijndael (AES), IDEA, Blowfish, RC4, CAST, GOST, SKIPJACK and SAFER, etc.

### 3.1.2 Asymmetric Key Cryptography
The public key cryptography is another name for the asymmetric key cryptography. In this type of cryptography a message or a block of bytes is converted with the help of well known public key. But the decryption takes place with the help of a private or a secret key. The currently used cryptosystem is shown in the Fig. 3.3.

![Encryption and Decryption Process Diagram](image)

Figure 3.3: An asymmetrical cryptosystem

It has been made public just in 1976 when Whitfield Diffie and Martin Hellman of Stanford University first introduced its concept. Diffie and Hellman were believed to be the pioneers of Asymmetric Key cryptography before the British Government Communications Headquarters (GCHQ), the equivalent of NSA, claimed to have conceived it years earlier and had been using it for years privately.

Some of the advantages of asymmetric key cryptography are:

- The private keys might be hidden here.
- The private and the public keys are kept same. It depends on the mode of utilization.
- The digital signatures are efficiently utilized in the public key schemes. The public verifications are utilized typically shorter one.
- In the usage of big networks, the private and the public keys should be shorter.

Drawbacks of the public key encryption scheme:

- Throughput rates for the most popular public-key encryption methods are several orders of magnitude slower than the best known symmetric-key schemes.
- Key sizes are typically much larger than those required for symmetric-key encryption, and the size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques.
- It is not a secure and well protected scheme as of the block ciphers. There are so many theoretical difficulties with the public key encryption systems.
This type of encryption has no history as of the symmetrical type of encryptions.

There are few popular asymmetric algorithms which in the utilization these are namely, Diffie-Hellman, RSA, DSA and Elliptic Curve Cryptography (ECC).

3.1.3 Hybrid Cryptography

As the name “Hybrid cryptography” depicts that it is a mixed up approach of the both types of cryptographies namely the symmetric one as well as the asymmetric one.

An asymmetrical key is utilized to encrypt the message or block of bytes and it is passed on the network with the help of an arbitrary key which is generated. This arbitrary key is used to encrypt the rest of the messages with the help of the symmetrical key encryption. In such type of cryptography both the features are utilized one is the pace and efficiency of the symmetrical algorithms and another one is the exchanging of private key securely. This type of cryptography is utilized in the Secure Socket Layer (SSL), SSH (Secure Socket Hashing), and PGP.

3.1.4 Cipher Modes for decryption and encryption operations

With the help of a Mode of operation it is described that a stream of data is encrypted using block ciphers, because block cipher algorithms themselves only describe what is done with a block of a fixed length. Data lengths are arbitrary, and to have secure encryption one has encrypt blocks differently, even though they have the same plain text input, thus avoiding dictionary and replay attacks. The simplest mode of encryption does not protect against these threats. This mode is called Electronic Codebook (ECB) and is regarded as insecure.

This is the only mode of operation which does not require an initialization vector. None of the modes of operation described in this section provides integrity protection, when they are used for encryption, meaning a change in the encrypted data either by error or an attack can go undetected. They can, however, provide integrity protection instead of encryption.

Initialization vector

An initialization vector (IV) is the initiator for the enciphering process for the initial block of message. The knowledge of the IV should be well understood for decryption process. However it is not to be recommended that it must be maintained as undisclosed like the encryption key. However, for encryption, the same IV should not be reused for encrypting different data, as it can be possible to detect patterns in the cipher text. The length of the IV must
also be large enough, to avoid IV collisions, which was a problem for security the Wired Equivalent Privacy 802.11 standard [50]. This made it possible to brute force attack the encryption of Wireless LANs using this form for encryption.

**Electronic Codebook (ECB)**

In this mode of operation, each block is encrypted in the same way. This is the simplest mode of operation to implement, and is easy to do in parallel because there are no inter-block dependencies. The encryption simply runs the cipher block algorithm on each block in the data; see Figure 3.4 for the encrypting procedure, and the decrypting procedure. This means that two identical plain texts will be encrypted to identical cipher texts. As mentioned it has a severe problem with security. We see in Figure 3.4 that it is possible to distinguish the original picture, and this shows that the encryption with ECB in some cases is not sufficient. One should however, note that even though ECB looks random, it is not a guarantee that the encryption is secure.

![Figure 3.4: ECB Encryption & Decryption (From Wikipedia)](image)

![Figure 3.5: CBC Encryption & Decryption (From Wikipedia)](image)

**CBC (Cipher Block Chaining)**
This is the mode of conversion that has been discovered at the IBM in nineteen hundred seventy six. Every simple text block is applied as a XOR operation to the preceding block-cipher key before the encryption. The initial plain text is XORed with the IV as seen in Figure 3.5. It is the very ordinary cryptographic mode. However there is a drawback that its encryption stage cannot be parallelized as efficiently as ECB, because of the inter-block dependencies of the mode of operation. To decrypt the data, each decrypted block must be XORed with the previous cipher key, or the IV for the first block as seen in Figure 3.5.

**CFB (Cipher Feedback)**

It is somewhat analogous to previous mode of operation. It first encrypts the IV, and XORs this encrypted data with the original message block for producing the cipher message. It then takes this cipher text, encrypts it and XORs it with the following plain text. It is the advantage of this mode above the CBC mode that the blocked cipher is simply utilized at the enciphering stage, and that there is no need to pad the message block to cipher block size multiplier. The encryption and decryption processes of Cipher Feedback (CFB) mode are displayed with the help of Figure 3.6.

![Figure 3.6: Cipher Feedback (CFB) Encryption & Decryption (From Wikipedia)](image)

**OFB (Output Feedback) mode**

It is a mode of operation, where the IV is enciphered, as well as the XORing is applied with the plain-text to yield enciphered text. It is known as a synchronous stream cipher. It has the advantages that it allows many error correcting codes to work, because it does not use the cipher text as input for the next encryption. The encryption and decryption processes of Cipher Feedback (CFB) mode are shown in the Figure 3.7.
Counter (CTR)

Counter (CTR) is also a stream cipher, but works quite different from the other modes of operation. It introduces a new term called nonce, that is similar to the IV, but has a changing element called a counter, in addition to a static element that remains the same throughout the encryption. CTR encrypts this nonce for every stage, with the counter element increased with one for each cipher block. The encrypted nonce is after that XORing is done with the help of plain-text for each cipher block, and this yields enciphered text as seen in Figure 3.8. Because of there are no inter-block dependencies, as seen in the figure CTR is fully parallelizable. In the above said figure we see how a block is decrypted, by decrypting the nonce and XORing it with the cipher text.

![Figure 3.7: Output Feedback (OFB) Encryption & Decryption (From Wikipedia)](image)

![Figure 3.8: Counter (CTR) Encryption & Decryption (From Wikipedia)](image)

3.1.5 Applications of Cryptography
There are basic two types of cryptographical approaches. These are symmetrical as well as asymmetrical cryptography. Cryptography is very practical; there are diverse nature’s applications in the daily use of ours. A classical application of cryptography is a system built out of the basic procedures. These types of systems may have many levels of complexities. The basic applications are like safe communication, identification, authentication, and secret data sharing. Additional complex applications are used in the e-commerce, certification, recovery of the keys, secure e-mail, secure file system etc.

3.2 Rijndael Algorithm (AES)

This algorithm was developed by the two Belgina scientists. They were Dr Joan Daemen and Dr Vincent Rijmen. The design was based on the concepts like to be very simple, and to be modular nature. The algorithm is a block cipher. Its block sizes have a lot of variations as one hundred and twenty eight (16 Bytes). The cipher keys are also diverse in their sizes. They may be of the lengths as 128 bits, 192 bits, and 256 bits. The key sizes in the bytes we can say are of 16 bytes, 24 bytes, and 32 bytes long. In this section of the text we will introduce the robust algorithm “The Rijndael Algorithm”. The flowcharts are drawn here.

3.2.1 Why is Rijndael Algorithm selected?

To be a simple algorithm, this concept is achieved with the help of not recognizing the typical operations. Here a very fast key scheduling is done. It increases the computational efficiency of the encryption procedure as well as the decryption procedure of the algorithm. The Rijndael algorithm is good choice for the development of a secure system whatever it is. This algorithm has the following features:

- **NIST Standard**: It is the United States FIPS standard.
- **Secure and efficient**: It provides very good performance with strong security and efficiency.
- **Hardware Implementations**: It can be manufactured on the (8 bits – 32 bits) CPUs (Processors).
- **Key size**: It is the most significant feature of the algorithm that it supports for 3 diverse key sizes (128 bits, 192 bits, and 256 bits). These are specified in the NIST FIPS standard.
**Design simplicity:** Its design is simple one.

**Resistance against all known attacks:** It has great resistance against all known attacks.

**Low Memory Requirements:** Rijndael requires few memory space (RAM). Because of this, it is the algorithm that can be in working with the limit space environments. The performance remains excellent. The pace and coding firmness has a wide range of platforms.

**Easy to understand and implement:** There are no complications in understanding of the algorithm. It is easily implemented in any computer language.

**Mode of operation:** It can be easily operated on any mode like CBC, ECB, CFB, OFB, OCB or counter modes of operations.

**Number of iterations:** There are many rounds in transforming the plaintext which makes increase in security.

**128 bits block size:** It works on the 128 bits individual block of data as an input.

### 3.2.2 Rijndael Algorithm

A US NIST had made selection of Rijndael Algorithm as one of the Federal Information Processing Standards (FIPS). The standard has given the name after their previous standard DES as the AES (Advanced Encryption Standard). The elaboration of the AES is shown here.

In the encryption process the block of data is transferred to non readable form that form is usually called the cipher text. Its reverse process is decryption in which this non readable data is transferred to the readable form that is basically the original one. A sequence of 16 bytes (128 bits) is the input for the Rijndael Algorithm which is taken from the plain text message or data. This sequence of data is encrypted with the help of a cipher key of size 128 bits (16 bytes). The length of cipher key, plain text and the cipher text is equal to the 128 bits (16 bytes) each. We keep the cipher key as a secret from the outsiders or intruders. All the sequential operations which are going to transfer the plaintext to the cipher text (non readable text) with the help of the secret key are known as the ciphers simply.

The 2-D (two-dimensional) arrays of bytes are utilized by the Rijndael algorithm to do all the conversion operations and it is known as the state. The 2-D array is shown with the help of [r,
c]’s. And it shows the name of the array. Here \( r \) denotes the row number and \( c \) denotes the column numbers. Here \( 4 \times 4 \) array represents the single state. Every block of the array is of size one byte.

Let us suppose that the single block of 128 bits has the bit positions as \( i_0 i_1 i_2 \ldots i_{126} i_{127} \). Here bytes in one state are shown as \( in_0 in_1 in_2 \ldots i_{14} i_{15} \), where \( in_0 = i_0 i_1 i_2 i_3 i_4 i_5 i_6 i_7 \), \( in_1 = i_8 i_9 i_{10} i_{11} i_{12} i_{13} i_{14} i_{15} \) \ldots \( in_{15} = i_{120} i_{121} i_{122} i_{123} i_{124} i_{125} i_{126} i_{127} \). All these sixteen bytes are shown in the Fig. 3.9.

Table 3.1: the Substitution-Box (S-Box) for the byte block \( xy \) [16]
The state array is used to translate the plain text (Input Bytes) into the cipher text (Output Bytes) by various processes of Rijndael Algorithm. A few of the mathematical equations are used to convert the input bytes in the state array and after that the state array is converted in to the output bytes. The equations are as follows:

\[ S[r, c] = in[r+4\cdot c] \quad \text{for} \quad 0 \leq r \leq 4 \quad \text{and} \quad 0 \leq c \leq 4 \]  
\[ Out(r+4\cdot c) = S[r, c] \]  

The state array is operated with the help of many functions like SubBytes ( ), InvSubBytes ( ), ShiftRows ( ), InvShiftRows ( ), MixColumns ( ), InvMixColumns ( ), AddRoundkey ( ), and InvAddRoundkey ( ) by the Rijndael algorithm. The normal functions takes place at the sender end at the time of encryption process and the reverse functions takes place at the recipient end at the time of decryption process. These functions are elaborated in the following sub headings.

**SubBytes ( ) Function**

![Figure 3.10: SubBytes ( ) Function with the S-Box to each byte of the State Array](image)

In this function, the transformations take place byte by byte in the state array with the help of concerned bytes in the S-Box which is described in the Table 3.1. The substitution of the bytes takes place after taking the corresponding rows and columns from the 2-D Substitution-Box. For instance, at the position 0, 1, there is an element 45 in the representation of the hexadecimal notations. Then it is exchanged with the new element which at the position 4th row and 5th column in the Substitution box Table 3.1. It is the 6e in the hexadecimal notations. This transformation is described with the help of Fig. 3.10. The Substitution box is generated with the help of the Golish Field (GF 2^8). This field has the nonlinear characteristics. The concerned
function offers nonlinearity for the cipher text bytes. The easy one algebraic characteristics are removed with the help of such transformations.

**InvSubBytes ( ) Function**

It is the reverse process that takes place on the end of the receivers. This is the inverse process of the SubBytes ( ) function conversion. Here another Substitution box is used that is called the reverse Substation-box that is described with the help of the Table 3.2. Here the state array bytes are exchanged with one by one bytes of the reverse Substitution box. The exchanging process is same as of the SubBytes ( ) function conversion. For instance, if state array position 1, 0 is the 38 in the hexadecimal notation then the corresponding reverse substitution box element value will be at the position 3\text{rd} row and the 8\text{th} column of the Table 3.2 that is 76 in the hexadecimal notation.

Table 3.2: The reverse Substitution box

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>52</td>
<td>09</td>
<td>6a</td>
<td>d5</td>
<td>30</td>
<td>36</td>
<td>a5</td>
<td>38</td>
<td>bf</td>
<td>40</td>
<td>a3</td>
<td>9e</td>
<td>81</td>
<td>f3</td>
<td>d7</td>
<td>fb</td>
</tr>
<tr>
<td>1</td>
<td>7c</td>
<td>e3</td>
<td>39</td>
<td>82</td>
<td>9b</td>
<td>2f</td>
<td>ff</td>
<td>87</td>
<td>34</td>
<td>8e</td>
<td>43</td>
<td>44</td>
<td>c4</td>
<td>de</td>
<td>e9</td>
<td>cb</td>
</tr>
<tr>
<td>2</td>
<td>54</td>
<td>7b</td>
<td>94</td>
<td>32</td>
<td>a6</td>
<td>e2</td>
<td>23</td>
<td>3d</td>
<td>ee</td>
<td>4e</td>
<td>95</td>
<td>0b</td>
<td>42</td>
<td>fa</td>
<td>e3</td>
<td>4e</td>
</tr>
<tr>
<td>3</td>
<td>08</td>
<td>2e</td>
<td>a1</td>
<td>66</td>
<td>28</td>
<td>d9</td>
<td>24</td>
<td>b2</td>
<td>76</td>
<td>5b</td>
<td>a2</td>
<td>49</td>
<td>6d</td>
<td>8b</td>
<td>d1</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>72</td>
<td>f8</td>
<td>f6</td>
<td>d4</td>
<td>86</td>
<td>68</td>
<td>98</td>
<td>16</td>
<td>d4</td>
<td>a4</td>
<td>5c</td>
<td>ee</td>
<td>5d</td>
<td>65</td>
<td>b6</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>6c</td>
<td>70</td>
<td>48</td>
<td>50</td>
<td>fd</td>
<td>ed</td>
<td>b9</td>
<td>da</td>
<td>5e</td>
<td>15</td>
<td>46</td>
<td>57</td>
<td>a7</td>
<td>8d</td>
<td>9d</td>
<td>84</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>d8</td>
<td>ab</td>
<td>00</td>
<td>8c</td>
<td>bc</td>
<td>d3</td>
<td>0a</td>
<td>f7</td>
<td>e4</td>
<td>58</td>
<td>05</td>
<td>b8</td>
<td>b3</td>
<td>45</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>d0</td>
<td>2e</td>
<td>1e</td>
<td>8f</td>
<td>ea</td>
<td>3f</td>
<td>0f</td>
<td>02</td>
<td>c1</td>
<td>af</td>
<td>bd</td>
<td>03</td>
<td>01</td>
<td>13</td>
<td>8a</td>
<td>6b</td>
</tr>
<tr>
<td>8</td>
<td>3a</td>
<td>91</td>
<td>11</td>
<td>4f</td>
<td>67</td>
<td>de</td>
<td>ea</td>
<td>97</td>
<td>f2</td>
<td>ef</td>
<td>ce</td>
<td>f0</td>
<td>b4</td>
<td>e6</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>96</td>
<td>ac</td>
<td>74</td>
<td>22</td>
<td>e7</td>
<td>ad</td>
<td>35</td>
<td>85</td>
<td>e2</td>
<td>f9</td>
<td>37</td>
<td>e8</td>
<td>1e</td>
<td>75</td>
<td>df</td>
<td>6e</td>
</tr>
<tr>
<td>a</td>
<td>47</td>
<td>f1</td>
<td>1a</td>
<td>71</td>
<td>1d</td>
<td>29</td>
<td>e5</td>
<td>89</td>
<td>6f</td>
<td>b7</td>
<td>62</td>
<td>0e</td>
<td>aa</td>
<td>18</td>
<td>be</td>
<td>1b</td>
</tr>
<tr>
<td>b</td>
<td>6e</td>
<td>56</td>
<td>3e</td>
<td>4b</td>
<td>e6</td>
<td>d2</td>
<td>79</td>
<td>20</td>
<td>9a</td>
<td>db</td>
<td>e0</td>
<td>fe</td>
<td>78</td>
<td>ed</td>
<td>5a</td>
<td>f4</td>
</tr>
<tr>
<td>c</td>
<td>1f</td>
<td>dd</td>
<td>a8</td>
<td>33</td>
<td>88</td>
<td>07</td>
<td>e7</td>
<td>31</td>
<td>b1</td>
<td>12</td>
<td>10</td>
<td>59</td>
<td>27</td>
<td>80</td>
<td>ec</td>
<td>5f</td>
</tr>
<tr>
<td>d</td>
<td>60</td>
<td>51</td>
<td>7f</td>
<td>a9</td>
<td>19</td>
<td>b5</td>
<td>4a</td>
<td>0d</td>
<td>2d</td>
<td>e5</td>
<td>7a</td>
<td>9f</td>
<td>93</td>
<td>e9</td>
<td>9e</td>
<td>ef</td>
</tr>
<tr>
<td>e</td>
<td>a0</td>
<td>e0</td>
<td>3b</td>
<td>4d</td>
<td>ae</td>
<td>2a</td>
<td>f5</td>
<td>b0</td>
<td>e8</td>
<td>eb</td>
<td>bb</td>
<td>3e</td>
<td>83</td>
<td>53</td>
<td>99</td>
<td>61</td>
</tr>
<tr>
<td>f</td>
<td>17</td>
<td>2b</td>
<td>04</td>
<td>7e</td>
<td>ba</td>
<td>77</td>
<td>d6</td>
<td>26</td>
<td>e1</td>
<td>69</td>
<td>14</td>
<td>63</td>
<td>55</td>
<td>21</td>
<td>0e</td>
<td>7d</td>
</tr>
</tbody>
</table>
In this type of conversion the same state array is taken into consideration. No shifting takes place in the first row means in the 0th row. The next all the three rows (1st, 2nd, and 3rd) are shifted by the offset values one, two, as well as three respectively. Shifting is done with the help of ShiftRows ( ) function are described with the help of the Fig. 3.11.

![Figure 3.11: ShiftRows ( ) function cycle](image)

This shifting is straightforward one that is done on the rows. In it the function achieves the left circulation shifting of the rows second, third, and the forth with the help of 1, 2, and 3 bytes respectively. The real Java language coding is as follows:

```java
void ShiftRows(byte[][] state)
{
    byte[] t = new byte[4];
    for (int r = 1; r < 4; r++)
    {
        for (int c = 0; c < Nb; c++)
        {
            t[c] = state[r][(c + r) % Nb];
        }
        for (int c = 0; c < Nb; c++)
        {
            state[r][c] = t[c];
        }
    }
}
```

**InvShiftRows ( ) Function**

It is the reverse (opposite) function of the ShiftRows ( ) function. As we know that there is no need to alter the first row (0th row). Now all the second, third, and forth rows are right cyclic shifted by the offset values 1, 2, and 3 respectively. This function’s working is displayed in the Figure 3.12.
In this function the rows are shifted in the reverse of the ShiftRows ( ) function. Now the circular right shift is done. The Java language coding is shown here.

```java
void InvShiftRows(byte[][] state)
{
    byte[] t = new byte[4];
    for (int r = 1; r < 4; r++)
    {
        for (int c = 0; c < Nb; c++)
        {
            t[(c + r)%Nb] = state[r][c];
        }
        for (int c = 0; c < Nb; c++)
        {
            state[r][c] = t[c];
        }
    }
}
```

**MixColumns ( ) Function**

As the name clues that it is the mixing of the columns and operation done. The input is the four bytes of the column of the specific state array and the output is the corresponding cipher text of four bytes. Every column of the state array is the modulo multiplication 4 + 1 by
the rigid polynomial ( ) that supposed to be the Golish Field (GF $2^8$). The working is displayed here with the help of the Figure 3.13.

The polynomial equation and the matrix multiplication are presented here.

$$a(x) = \{03\} x^3 + \{01\} x^2 + \{01\} x + \{02\}$$  \hspace{1cm} (3.3)

$$s(x) = a(x) \otimes s(x)$$  \hspace{1cm} (3.4)

$$\begin{bmatrix}
S'_{0,c} \\
S'_{1,c} \\
S'_{2,c} \\
S'_{3,c}
\end{bmatrix} = \begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02
\end{bmatrix} \begin{bmatrix}
S_{0,c} \\
S_{1,c} \\
S_{2,c} \\
S_{3,c}
\end{bmatrix}$$  \hspace{1cm} for \hspace{0.5cm} 0 \leq c < 4  \hspace{1cm} (3.5)

**InvMixColumns ( ) Function**

This function is the opposite of the MixColumn ( ) function that is processed at the recipients end. It is also operated on the state array. Four bytes are the input for it and the same number of bytes is the output size. The same procedure is followed here as of the MixColumn ( ) function. The rigid polynomial $-1 ( )$ takes place on the every column of the state array to produce the result of the multiplicative modulo $4 + 1$. The polynomial is above the Galoish Field ($2^8$). The equations for the polynomials and the matrix multiplications are presented here.

$$a^{-1}(x) = \{0b\} x^3 + \{0d\} x^2 + \{09\} x + \{0e\}$$  \hspace{1cm} (3.6)

$$s(x) = a(x) \otimes s(x)$$  \hspace{1cm} (3.7)

$$\begin{bmatrix}
S'_{0,c} \\
S'_{1,c} \\
S'_{2,c} \\
S'_{3,c}
\end{bmatrix} = \begin{bmatrix}
0e & 0b & 0d & 09 \\
09 & 0e & 0b & 0d \\
0d & 09 & 0e & 0b \\
0b & 0d & 09 & 0e
\end{bmatrix} \begin{bmatrix}
S_{0,c} \\
S_{1,c} \\
S_{2,c} \\
S_{3,c}
\end{bmatrix}$$  \hspace{1cm} for \hspace{0.5cm} 0 \leq c < 4.  \hspace{1cm} (3.8)

**AddRoundKey ( ) Function**

A small key is obtained from the main secret key for every round of the encryption process of the Rijndael Algorithm. It is done with the help of the key expending algorithm. The small key’s length is of sixteen bytes. This small key is further subdivided into the collection of the four words. The bitwise exclusive-OR operation takes place on very column of the state array. Its range is $0 \leq \text{range} \leq 4$. The complete procedure is described here in the Figure 3.14.
The mathematical presentation of the AddRoundKey ( ) conversion is described in the equation 3.9.

\[
[s_0,c,s_1,c,s_2,c,s_3,c] = [s_0,c,s_1,c,s_2,c,s_3,c] \oplus [w_{4\cdot round+c}] \text{ for } 0 \leq c \leq 4, \tag{3.9}
\]

The \( w \) is the expended key portion here. It is exclusive-ORed against the state matrix \( N_{r+1} \) times. A state has \( 4 \times Nb \) bytes. Every byte of the expended key is utilized at once. The size of the expended key is \( 4 \times Nb \times (N_{r+1}) \) bytes. This key is utilized byte with the byte from the lower to higher indices. The counting of the bytes is useless. These are used from the \( w \), the expended key. The Java language code is described here.

```java
void AddRoundKey(byte[][] state) {
    for (int c = 0; c < Nb; c++)
        for (int r = 0; r < 4; r++)
            state[r][c] = state[r][c] ^ w[wCount++];
}
```

InvAddRoundkey ( ) Function

This function has the opposite working from the above mentioned function named as AddRoundKey. Its functioning is totally in reverse order from the above mentioned function. Java language program coding is as follows:
3.2.3 Expansion of Key

```java
void InvAddRoundKey(byte[][] state)
{
    for (int c = Nb - 1; c >= 0; c--)
        for (int r = 3; r >= 0; r--)
            state[r][c] = state[r][c] ^ w[--wCount];
}
```

Figure 3.15: Key expansion flowchart.

Encipher and decipher keys are expanded for the different rounds of cryptographical algorithm. The addroundkey function is utilized so many times for the encryption and the
decryption processes. For each round a different key is utilized. A sequence of novel keys is used from the previous one. So the expansion key routine is used for producing the novel keys which provide the more security to the algorithm. In the figure 3.15 the flowchart of the expansion of the key is shown. There are in total ten keys which are derived from the basic key. The constant of the each round is the multiplication of two.

A temporary word is maintained for storing the most recent 4 bytes of the key. On this word the left shifting is performed. It is done like the state array shifting.

The process of substituting is same as of the SubByte function operations. The ex-oring is done on the maintained word and also to the constant round. The exclusive oring takes place on the cipher key of the temp word. The novel 128-bt key is the result that is called the key after expansion.

The novel key is utilized for enciphering key for the following round. The next key is obtained with the help of the same procedure which is use in the Rijndael algorithm.

### 3.2.4 Rijndael Algorithm Encryption and Decryption Flowcharts

![Flowchart](image)

Figure 3.16: RIJNDAEL Algorithm (AES) encryption and decryption flowcharts.

The flowcharts for the processes encryption as well as the decryption are shown in the figure 3.16. The encryption flowchart shows the working of the sender’s end process. And the
decryption process flowchart show the working at the recipient end. All of the functions which are described in the flowchart are explained in the previous section of this chapter.